MATH 7339 - Machine Learning and Statistical Learning Theory 2

Section 1. Statistical Learning

1. Overview
2. Bias –Variance Trade-off
3. Examples
4. Curse of dimension

➢ Set up

Variables:

Input (feature/predictor): $\vec{x} = (x_1, \dots, x_d) = \begin{bmatrix} x_1 \\ \vdots \\ x_d \end{bmatrix} \in \mathcal{X} = \mathbb{R}^d$

Output (label): $y \in \mathcal{C} \subseteq \mathbb{R}$

Note here y is an outcome we wish to predict from $\vec{x}$.

Assumption: Whenever we measure a pair $(\vec{x}, y)$, they are an example/sample of a pair of **Random Variables** (R.V.) $\left(\vec{X}, Y\right)$ with some fixed underlying distributions.

We want to 'Learn' relation between $\vec{x}$ and y from the training data $\mathcal{D} = \left\{\left(\vec{x}^{(i)}, y^{(i)}\right),\ i = 1, \dots n\right\}$.

➢ Statistical learning and machine learning:

- Prediction of response y from predictors $x_1, x_2, \ldots, x_p$. (Regressions and Classifications)
- Identification of 'important' variables affecting outcome. (Feature selection)
- Characterizing associations among variables.
- Statistical learning is less algorithm based than standard machine learning, which is more algorithm based. We will do both.

More formal definitions:

Assume we have variables $\vec{x} \in \mathcal{X}$ and $y \in \mathcal{C}$ in some respective domains $\mathcal{X}$ and $\mathcal{C}$.

- A **learning algorithm** is a **rule** that takes a training set $\mathcal{D} = \left\{\left(\vec{x}^{(i)}, y^{(i)}\right), i = 1, \ldots n\right\}$ as input and produces as its output a predictor function $y = \hat{f}(\vec{x})$ that for every $\vec{x}$ value predicts a y value.

- Given predictors $\vec{x} \in \mathcal{X} \subset \mathbb{R}^d$ and outputs $y \in \mathcal{C}$, a **(probability) model** of their relationship is a joint probability distribution $\boldsymbol{P}$ on the pair $\left(\vec{X}, Y\right) \in \mathbb{R}^d \times \mathcal{C}$ of a random vector $\vec{X} \subset \mathbb{R}^d$ and random variable $Y \in \mathcal{C}$.

- Assume there is an unknown probabilistic relationship between a predictor $\vec{x} \in \mathcal{X} \subset \mathbb{R}^d$ and an output $y \in \mathcal{C}$. We define a **(function) model** of this relationship to be a single function $f: \mathcal{X} \to \mathcal{C}$ or a class $\mathcal{F}$ of such functions one of which is assumed to give a 'good' prediction $y \in \mathcal{C}$ from $\vec{x} \in \mathcal{X} \subset \mathbb{R}^d$.

➢ Supervised v.s. Unsupervised

❑ Types of statistical learning:

1. Supervised learning
2. Unsupervised learning

Example:

- Supervised case: $(y, \vec{x}) \rightarrow P(y|\vec{x})P(\vec{x})$
- Unsupervised case: $(\vec{x}) \rightarrow P(\vec{x})$

➢ Regression v.s. Classification

❑ Two data types for the outcome y:

1. $y \in \mathbb{R}$ is **continuous**, quantitative. (regression)

2. $y \in \{1, \ldots, K\}$ is **discrete**, qualitative. (classification)

Regression and classification are often in the same theoretical context.

In both cases, a (probability) model fixed joint probability distribution.

$$\left(\vec{X}, Y\right) \sim P(\vec{x}, y)$$

Specifically, we will say that for a (measurable) set $A \subset \mathbb{R}^d \times \mathcal{C}$, the **probability that** $\left(\vec{X}, Y\right) \in A$ is

$$P(A) \coloneqq P\left(\left(\vec{X}, Y\right) \in A\right)$$

In regression case $y \in \mathbb{R}$ we often assume that the measure P has a joint probability density function.

$$p(\vec{x}, y) = p(x_1, x_2, \ldots, x_d, y)$$

For a (measurable) set A

$$P\left((\vec{X}, Y) \in A\right) = \int_A p(\vec{x}, y)\, d\vec{x} dy$$

We typically assume **a regression model** has the form**:**

$$y = h(\vec{x}) + \epsilon$$

Here $h(\vec{x})$ best predicts y, but with some small ϵ **random error.**

We cannot control the error ϵ, but want the best choice of $h(\vec{x})$ to predict y from $\vec{x}$.

➢ Parametric vs. non-parametric methods:

Mathematical modeling(Statistical learning) infers (from training data $\mathcal{D} = \{(\vec{x}^{(i)}, y^{(i)}),\ i = 1, \dots n\}$) a predictor function

$$\hat{h}(\vec{x}) \approx h(\vec{x})$$

for the relationship $y^{(i)} = h(\vec{x}^{(i)})$. We write the estimate $\hat{y} = \hat{h}(\vec{x})$

Suppose the actual relationship is $y^{(i)} = h(\vec{x}^{(i)}) + \epsilon_i$, where ϵ_i is a random error. Usually, y can only be approximately predicted from $\vec{x}$, not perfectly predicted.

Definition: A **parametric inference method** infers that the approximating function $\hat{f}(\vec{x})$ is determined by a finite number of parameters

$$\vec{\theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_d \end{bmatrix}$$

Definition: A **non-parametric inference method** estimates f without parameters, or with an infinite number of parameters.

Parametric method (Formula, fast, less data)
Examples:

1. Linear regression $h(\vec{x}) = \vec{\theta}^T \vec{x} = \theta_0 + \theta_1 x_1 + \cdots + \theta_d x_d$
2. Ridge/Lasso
3. Logistic regression
4. Linear Discriminant Analysis (LDA)/ QDA
5. Basic Support vector machine(SVM)
6. Perceptron
7. Naive Bayes
8. Basic Neural Network

Non-Parametric method (Fewer assumptions, Good fit, Accurate)
Examples:

1. K-nearest neighbors(kNN)
2. Locally weighted linear regression
3. Decision trees (Random forest.)
4. Support vector machine(SVM) with kernel.

Along the study of statistical learning theory, we will review and use the above models as examples.

➢ Notations

Assume a set of random variables U_i forming a random vector $\vec{U}$ and additional variables $\vec{v}$.

$$\vec{U} = \begin{bmatrix} U_1 \\ U_2 \\ \vdots \\ U_n \end{bmatrix} \qquad \vec{v} = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_m \end{bmatrix}$$

Consider any function

$$h\left(\vec{U}, \vec{v}\right) = h(U_1, \ldots, U_n, v_1, \ldots, v_m)$$

We define $E_U\left[h\left(\vec{U}, \vec{v}\right)\right]$ to be the **expectation** of $h\left(\vec{U}, \vec{v}\right)$ with respect to the variables $\vec{U}$ only, not including variables $\vec{v}$.

Thus, variables $\vec{v}$ held constant while expectation is computed.

➢ Assumptions/models

We assume **a regression model:**

$$y^{(i)} = h\left(\vec{x}^{(i)}\right) + \epsilon_i$$

Here $h(\vec{x})$ depends on some (finite or infinite) collection of parameters $\vec{\theta}$.

Assume errors ϵ_i are random variables of the form

$$\epsilon_i \sim \text{Normal}(0, \sigma^2)$$

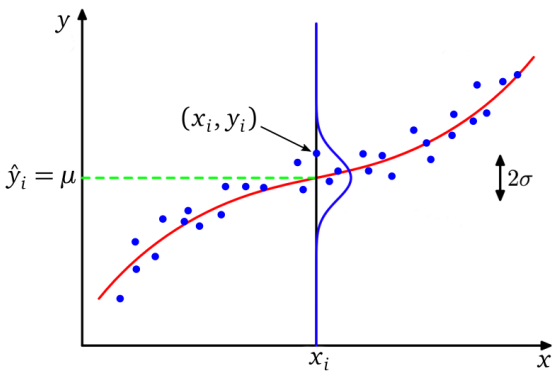
Given specific value $\vec{x}$ for R.V. $\vec{X}$, the expected value

$$E\left(Y \middle| \vec{X} = \vec{x}\right) = E\left(h\left(\vec{X}\right) + \epsilon \mid \vec{X} = \vec{x}\right)$$

$$= E\left(h\left(\vec{X}\right) \middle| \vec{X} = \vec{x}\right) + E\left(\epsilon \mid \vec{X} = \vec{x}\right)$$

$$= 0 + h(\vec{x})$$

$$= h(\vec{x})$$



We wish to predict the outcome $y^{(0)}$ given input (test point) $\vec{x}^{(0)}$.

$$y^{(0)} = h\left(\vec{x}^{(0)}\right) + \epsilon$$

Denote our prediction of $y^{(0)}$ to be $\hat{y}^{(0)}$

We can view $y^{(0)}$ itself be a random variable due to the noise component ϵ_i

The training data set $\mathcal{D}$ itself is made of random variables (before we see $\mathcal{D}$) because of noise ϵ_i in each $y^{(i)}$ and also randomness in the choice of training points $x^{(i)}$.

➢ Expected Prediction Error

Assume we have an algorithm for estimating $h(\vec{x}^{(0)})$ for some test point $\vec{x}^{(0)}$

We denote the resulting estimator

$$\hat{y}^{(0)} = \hat{h}(\vec{x}^{(0)}) \approx h(\vec{x}^{(0)})$$

The decision theory also needs a **loss function:**

$$L(Y, h(\vec{X}))$$

For regression, we often choose **Squared error loss or Absolute error**

$$L\left(Y, h(\vec{X})\right) := \left(Y - h(\vec{X})\right)^2 \text{ or } \quad L\left(Y, h(\vec{X})\right) := |Y - h(\vec{X})|$$

Suppose a (probability) model is the joint probability distribution.

$$(\vec{X}, Y) \sim P(\vec{x}, y)$$

We assume joint probability density function $p(\vec{x}, y)$ is fully known for now.

Expected Prediction Error

$$EPE(h) := E_{Y,\vec{X}}\left[L\left(Y, h(\vec{X})\right)\right]$$

Conditioning on $\vec{X}$,

$$EPE(h) := E_{Y,\vec{X}}\left[L\left(Y, h(\vec{X})\right)\right]$$

$$= E_{\vec{X}} E_{Y|\vec{X}}\left[L\left(Y, h(\vec{X})\right)|\vec{X}\right]$$

So our best model is to minimize pointwise

$$\hat{h}(\vec{x}) = \underset{c}{\operatorname{argmin}}\, E_{Y|\vec{X}}\left[L(Y, c)|\vec{X} = \vec{x}\right]$$

Here we use the probability theorem-
law of iterated expectations:

$$E(A) = E_B\left[E_{A|B}\left[A|B\right]\right]$$

Another important theorem is
law of total variance:

$$Var(A) = E[Var(A|B)] + Var(E(A|B))$$

- In the case **Squared error loss,** $L\left(Y, h(\vec{X})\right) \coloneqq \left(Y - h(\vec{X})\right)^2$, the solution is

$$\hat{h}(\vec{x}) = \underset{c}{\operatorname{argmin}} E_{Y|\vec{X}}\left[(Y-c)^2 | \vec{X} = \vec{x}\right]$$

Here c is the variable.

$$= \underset{c}{\operatorname{argmin}} E_{Y|\vec{X}}\left[Y^2 - 2cY + c^2 | \vec{X} = \vec{x}\right] = E[Y|\vec{X} = \vec{x}]$$

- In the case of **Absolute error,** $L\left(Y, h(\vec{X})\right) \coloneqq |Y - h(\vec{X})|$, the solution is

$$\hat{h}(\vec{x}) = \underset{c}{\operatorname{argmin}} E_{Y|\vec{X}}\left[|Y-c| | \vec{X} = \vec{x}\right]$$

$$= median[Y|\vec{X} = \vec{x}]$$

This is more robust for outliers.

Classification Example:

Binary case when $Y = 0, 1$:

$$Y = \begin{cases} 1 & \text{if class 1} \\ 0 & \text{if class 2} \end{cases}$$

It makes sense to use

$$\hat{h}(\vec{x}) = E[Y|\vec{X} = \vec{x}]$$

$$= P\left(class\ 1\ \middle|\vec{X} = \vec{x}\right) \cdot 1 + P\left(class\ 2\ \middle|\vec{X} = \vec{x}\right) \cdot 0$$

$$= P\left(Y = 1\ \middle|\vec{X} = \vec{x}\right) \cdot 1 + P\left(Y = 0\ \middle|\vec{X} = \vec{x}\right) \cdot 0$$

$$= P\left(Y = 1\ \middle|\vec{X} = \vec{x}\right) \in \mathbb{R}$$

A class prediction function (**classifier**) is

$$\hat{f}(\vec{x}) = \begin{cases} 1 & \text{if } \hat{h}(\vec{x}) > 0.5 \\ 0 & \text{if } \hat{h}(\vec{x}) < 0.5 \end{cases}$$

In the multiclass case when $Y = 1, \ldots, K$

$$\hat{h}(\vec{x}) = \underset{f(X)}{\operatorname{argmin}} E_{Y|\vec{X}}\left[L(Y, h(X))|\vec{X} = \vec{x}\right]$$

$$= \underset{k}{\operatorname{argmin}} E_{Y|\vec{X}}\left[L(Y, k)|\vec{X} = \vec{x}\right]$$

$$= \underset{k}{\operatorname{argmin}} \sum_{y=1}^{K} L(y, k) P\left(Y = y \middle| \vec{X} = \vec{x}\right) \qquad \text{Here } k = 1, \ldots, K$$

For example, use the $0 - 1$ loss function

$$L(y, k) = \begin{cases} 0 & k = y \\ 1 & k \neq y \end{cases}$$

In this case

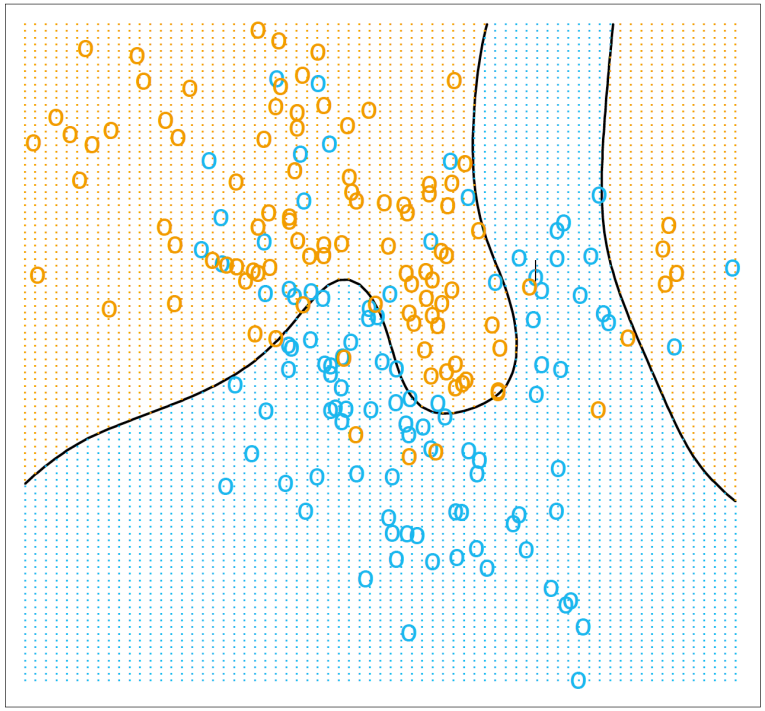
$$\hat{h}(\vec{x}) = \underset{k}{\operatorname{argmin}} \sum_{y \neq k} P\left(y \middle| \vec{X} = \vec{x}\right) = \underset{k}{\operatorname{argmin}}\, 1 - P\left(Y = k \middle| \vec{X} = \vec{x}\right)$$

$$= \underset{k}{\operatorname{argmax}}\, P\left(Y = k \middle| \vec{X} = \vec{x}\right)$$

Decision Boundary

The decision boundary between class j and class k is

$$\{\vec{x} \in \mathbb{R}^d \mid P(y = j|\vec{X} = \vec{x}) = P(y = k|\vec{X} = \vec{x})\}$$



The optimal Bayes decision boundary for simulation example. Generating density is known for each class.

➢ Bias –Variance Trade-off

We assume **a regression model,** with iid ϵ_i:

$$y^{(i)} = h\left(\vec{x}^{(i)}\right) + \epsilon_i$$

Assume there is an estimate $\hat{h}(\vec{x})$, and denote the resulting estimator at a test point $\vec{x}^{(0)}$ as

$$\hat{y}^{(0)} = \hat{h}\left(\vec{x}^{(0)}\right) \approx h\left(\vec{x}^{(0)}\right)$$

Definition: The **Expected Prediction(/test) Error (EPE) in this estimate (at a fixed test point $\vec{x}^{(0)}$)** is an average over both $y^{(0)}$ (the actual value of y at $\vec{x}^{(0)}$) and also the **entire** training set $\mathcal{D}$.

$$EPE\left(\vec{x}^{(0)}\right) := E_{y^{(0)},D}\left[L\left(y^{(0)}, \hat{h}\left(\vec{x}^{(0)}\right)\right)\right]$$

We seek **full expected prediction error**, averaging over all training sets $\mathcal{D} = \left\{\left(\vec{x}^{(i)}, y^{(i)}\right),\ i = 1, \ldots n\right\}$) and over all possible values $Y = y^{(0)}$ at a **fixed** text point $\vec{x}^{(0)}$.

Note both $\mathcal{D}$ and $y^{(0)}$ treated as random variables, from some underlying distribution.

Mathematical Decomposition of Expected prediction(test) error at fixed $\vec{x}^{(0)}$

$$EPE(\vec{x}^{(0)}) := E_{y^{(0)},D}\left[\left(y^{(0)} - \hat{y}^{(0)}\right)^2\right]$$

$$= E_{y^{(0)}}\, E_D\left[\left(y^{(0)} - \hat{y}^{(0)}\right)^2\right]$$

$$= E_{y^{(0)}}\, E_D\left\{\left[\left(y^{(0)} - E_{y^{(0)}}\left[y^{(0)}\right]\right) + \left(E_{y^{(0)}}\left[y^{(0)}\right] - E_D\left[\hat{y}^{(0)}\right]\right) + \left(E_D\left[\hat{y}^{(0)}\right] - \hat{y}^{(0)}\right)\right]^2\right\}$$

based on test set

based on training set

We can show that all cross terms in the squared expression disappear above.

$$E_{y^{(0)}} E_D \left[\left(y^{(0)} - E_{y^{(0)}}\left[y^{(0)}\right] \right) \left(E_D\left[\hat{y}^{(0)}\right] - \hat{y}^{(0)} \right) \right]$$

$$= E_{y^{(0)}} \left[\left(y^{(0)} - E_{y^{(0)}}\left[y^{(0)}\right] \right) E_D\left(E_D\left[\hat{y}^{(0)}\right] - \hat{y}^{(0)} \right) \right] = 0$$

Note $y^{(0)}$ and $\hat{y}^{(0)}$ are independent since their ϵ_i are independent and only $\hat{y}^{(0)}$ depends on D.

$$EPE\left(\vec{x}^{(0)}\right) = E_{y^{(0)}} \left[y^{(0)} - E_{y^{(0)}}\left[y^{(0)}\right] \right]^2 + \left[E_{y^{(0)}}\left[y^{(0)}\right] - E_D\left[\hat{y}^{(0)}\right] \right]^2$$

$$+ E_D\left[E_D\left[\hat{y}^{(0)}\right] - \hat{y}^{(0)} \right]^2$$

1. variance of $y^{(0)}$
2. constants
3. variance of $\hat{y}^{(0)}$ ($E_D\left[\hat{y}^{(0)}\right]$ is constant, and $\hat{y}^{(0)}$ depends only on D)

$$= Var_{y^{(0)}}\left(y^{(0)}\right) + \left[E_{y^{(0)}}\left[y^{(0)}\right] - E_D\left[\hat{y}^{(0)}\right]\right]^2 + Var_D\left(\hat{y}^{(0)}\right)$$

$$= \sigma^2 + \left[h(x^{(0)}) - E_D\left[\hat{y}^{(0)}\right]\right]^2 + Var_D\left(\hat{y}^{(0)}\right)$$

$$= \sigma^2 + [Bias]^2(\hat{y}^{(0)}) + Var_D\left(\hat{y}^{(0)}\right)$$

= Unavoidable error + bias^2 +variance

Note:

$$E_{y^{(0)}}\left[y^{(0)}\right] = E_{y^{(0)}}\left[h(x^{(0)}) + \epsilon\ \right] = h(x^{(0)})$$

1. The Unavoidable error/noise σ^2 term is always there.
2. The bias term estimates how 'on target' the method is on the average.
3. The $Var_D\left(\hat{y}^{(0)}\right)$ term represents sensitivity of the estimator to noise ϵ in training set D. Also to the choice of training points $\vec{x}^{(i)}$
4. This is a classic example of bias-variance tradeoff

Definition: We define the **bias** of an estimator $\hat{\vec{\theta}}$ to be the average error

$$Bias\left(\hat{\vec{\theta}}\right) := \left\| E_{\vec{\theta}}(\vec{\theta}) - E_D\left(\hat{\vec{\theta}}\right) \right\|$$

between the parameter $\vec{\theta}$ and its estimate $\hat{\vec{\theta}}$ based on a training set.

Note:

- $E_{\vec{\theta}}(\vec{\theta})$ is the average value of parameter $\vec{\theta}$ in a data universe.
- $E_D\left(\hat{\vec{\theta}}\right)$ is the average value of the estimator $\hat{\vec{\theta}}$ from a training set D, over the universe of training sets D.

We have defined the average error between actual value $y^{(0)}$ and its estimate $\hat{y}^{(0)}$

$$Bias(\hat{y}^{(0)}) := \left| E_{y^{(0)}}(y^{(0)}) - E_D(\hat{y}^{(0)}) \right|$$

Remarks:

The bias-variance decomposition may provide some interesting insights into the model complexity issue from a **frequentist** perspective.

It is of limited practical value, because the bias-variance decomposition is based on averages with respect to ensembles of data sets, whereas in practice we have only the single observed data set.

If we had a large number of independent training sets of a given size, we would be better off combining them into a single large training set, which of course would reduce the level of over-fitting for a given model complexity.

In practice, we will use Cross-Validation to estimate the test error.

Bayesian treatment of linear basis function models, which not only provides powerful insights into the issues of over-fitting but which also leads to practical techniques for addressing the question model complexity.

Example: k-NN method:

A k-Nearest Neighbor (k-NN) model, which is local, unstructured, and non-parametric:

$$\hat{h}(\vec{x}^{(0)}) = \frac{1}{k} \sum_{\vec{x}^{(i)} \in \mathcal{N}_0} y_i$$

where the k training observations that are closest to x_0, represented by $\mathcal{N}_0$

For the k-NN method, intuitively, for larger number of neighbors k, we average more points in training set D , giving more stable results. Thus, we have variance decreasing with k gets larger.

On the other hand, for increasing k we estimate $h(\vec{x}^{(0)})$ from training points $\vec{x}^{(i)}$ that may be far from test point $\vec{x}^{(0)}$. Thus, the bias increases with k gets larger.

Thus, we get decreasing variance along with increasing bias as k gets larger.

Assume $Y = h(\vec{X}) + \epsilon$, where $\epsilon \sim \text{Normal}(0, \sigma^2)$

$$\hat{y}^{(0)} = \hat{h}(\vec{x}^{(0)}) = \frac{1}{k} \sum_{\vec{x}^{(i)} \in \mathcal{N}_0} y^{(i)}$$

$=$ Average of values of $y^{(i)}$ for k points $\vec{x}^{(i)}$ nearest to $\vec{x}^{(0)}$.

So,

$Var_D(\hat{y}^{(0)}) =$ variance of average of k independent variables $y^{(i)}$

$$= \frac{\text{variance of single } y^{(i)}}{k}$$

$$= \frac{\sigma^2}{k}$$

Expected prediction error at $\vec{x}^{(0)}$ is

unavoidable bias^2 variance

$$\mathrm{EPE}\left(\vec{x}^{(0)}\right) = \sigma^2 + \left[h(x^{(0)}) - E_D\left[\hat{y}^{(0)}\right]\right]^2 + Var_D\left(\hat{y}^{(0)}\right)$$

$$= \sigma^2 + \left[h(x^{(0)}) - E_D\left[\frac{1}{k}\sum_{x_i \in \mathcal{N}_0} y^{(i)}\right]\right]^2 + \frac{\sigma^2}{k}$$

bias increases with k variance decreases with k

Here the trade-off in bias versus variance is evident!

Example: linear regressions:

A linear model, which is global, structured, and parametric:

$$\hat{y} = \hat{h}_\theta(\vec{x}\,) = \vec{x}^T\vec{\theta} = \theta_0 + \theta_1 x_1 + \cdots + \theta_d x_d$$

If this model is correct (i.e., $y = \vec{x}^T\vec{\theta} + \epsilon$) and we use our regression method, can show bias term in previous calculation disappears.

But if model is not quite correct, we pay a bias penalty (bias is non-zero) for the imposition of our linear structure assumptions. For example, Ridge regression, Lasso regression.

In Machine Learning, we will be working on these themes.

Assume that the **true** relationship between R.V. $\vec{X}$ and Y is exactly equation is

$$Y = h(\vec{X}) + \epsilon = \vec{X}^T\vec{\theta} + \epsilon$$

where $\epsilon \sim \text{Normal}(0, \sigma^2)$, which does not depend on $\vec{X}$.

Using a training set $\mathcal{D}$ of independent samples $\{(\vec{x}^{(i)}, y^{(i)}),\ i = 1, \ldots n\}$ of the underlying R.V. $(\vec{X}, Y)$. For these samples, we have

Given specific value $\vec{x}$ for R.V. $\vec{X}$, the expected value

$$E(Y|\vec{X} = \vec{x}) = \cdots = h(\vec{x})$$

Under the assumption, the maximum likelihood estimator of of $\vec{\theta}$ is given by the least-squares solution:

$$\hat{\vec{\theta}} = (X^TX)^{-1}X^T\vec{y}$$

where $(\text{X}, \vec{y})$ is sample data matrix and label vector and suppose $\text{rank}(X) = d$, full column rank.

Linear Algebra understanding of linear model:

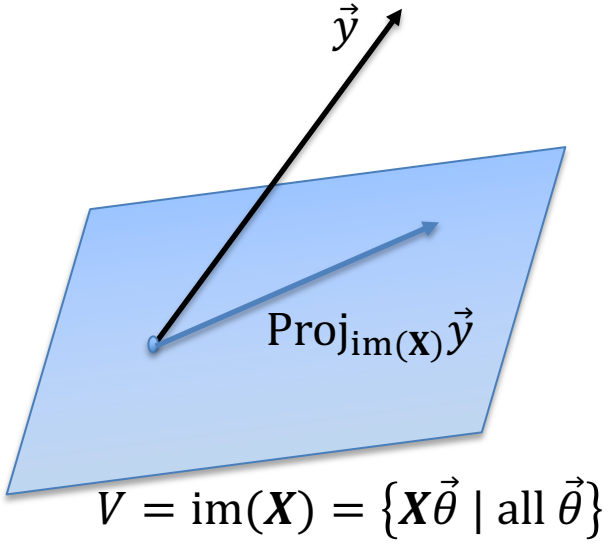
Assume Cost/Loss Function: $L(\vec{\theta}) = ||h_\theta(\boldsymbol{X}) - \vec{y})||^2$ by the Euclidean distance on $\mathbb{R}^n$ induced by dot product.

The optimization $\hat{\vec{\theta}} = \underset{\vec{\theta}}{\operatorname{argmin}} L(\vec{\theta})$ is solved by

$$\boldsymbol{X}\hat{\vec{\theta}} = \text{Proj}_{\text{im}(\mathbf{X})}\vec{y}$$

Equivalently, if $\text{rank}(X) = d$, then

$$\hat{\vec{\theta}} = (X^T X)^{-1} X^T \vec{y}$$



The prediction $\hat{y} = X\hat{\vec{\theta}} = \text{Proj}_{\text{V}}\, \vec{y} = H\vec{y}$, where $H = X(X^T X)^{-1} X^T$

The full regression is equivalent to stepwise orthogonal regression on each x_i.

- **Best Linear Unbiased Estimator (BLUE).**

The **mean** and **covariance** matrix of the least-squares estimate $\hat{\vec{\theta}} = (X^T X)^{-1} X^T \vec{y}$ are

$$E(\hat{\vec{\theta}}) = \vec{\theta}. \text{ (unbiased)}$$

$$\text{Cov}(\hat{\vec{\theta}}) = \sigma^2 (X^T X)^{-1}$$

Gauss-Markov Theorem. Among all *linear, unbiased* estimates for $\vec{\theta}$, the least-squares estimate $\hat{\vec{\theta}} = (X^T X)^{-1} X^T \vec{y}$ has the *smallest variance*.

Least squares estimator is the Best Linear Unbiased Estimator (**BLUE**).

There may still exists **biased estimators** with a smaller mean squared error.
That is, we may be able to **trade** a small increase in bias for a large reduction in variance.

$$\mathrm{Cov}\left(\hat{\vec{\theta}}\right) = \sigma^2 (X^T X)^{-1}$$

Note that one way for the variance above to increase is to have a near-singular matrix $(X^T X)$ (i.e., with some very small positive eigenvalues), so that $(X^T X)^{-1}$ is large and unstable.

This corresponds to some components x_i and x_j of feature vector $\vec{x}$ being highly correlated with each other (almost identical), so columns of X are almost proportional.

Thus such a pair of almost identical variables x_i and x_j can have very large coefficients θ_i and θ_j that simply cancel each other out.

Ridge regression was originally proposed as a remedy to this problem, by controlling such excessively large coefficients

Ridge Regression

We will assume also that data $\mathcal{D} = (X, \vec{y})$ here are centered, i.e. the mean $E(X) = \vec{0}\ and\ E(\vec{y}) = 0$.

$$h(\vec{x}) = \vec{\theta}^T \vec{x} = \theta_1 x_1 + \cdots + \theta_d x_d$$

Ridge regression minimize cost function:

$$J^{Ridge}(\vec{\theta}\,) = \sum_{i=1}^{n} \left(y^{(i)} - h_\theta(\vec{x}^{(i)})\right)^2 + \lambda \sum_{j=1}^{d} {\theta_j}^2$$

$$= \left(X\vec{\theta} - \vec{y}\right)^T \left(X\vec{\theta} - \vec{y}\right) + \lambda \vec{\theta}^T \vec{\theta}$$

Calculate $\nabla_{\vec{\theta}} J = 0$, we get

$$\vec{\theta} = (X^T X + \lambda I)^{-1} X^T \vec{y}$$

Expected prediction error for linear regression

$$\mathrm{EPE}\left(\vec{\boldsymbol{x}}^{(\mathbf{0})}\right) = E_{y^{(0)},D}\left[\left(y^{(0)} - \hat{y}^{(0)}\right)^2\right]$$

$$= \sigma^2 + \left[h(x^{(0)}) - E_D\left[\hat{y}^{(0)}\right]\right]^2 + E_D\left[E_D\left[\hat{y}^{(0)}\right] - \hat{y}^{(0)}\right]^2$$

$$= \sigma^2 + [Bias]^2(\hat{y}^{(0)}) + Var_D\left(\hat{y}^{(0)}\right)$$

Assume $y^{(0)} = h\left(\vec{x}^{(0)}\right) + \epsilon = \vec{x}^{(0)^T}\vec{\theta} + \epsilon$. Here $\vec{\theta}$ is the true value.

$$\hat{y}^{(0)} = \hat{h}\left(\vec{x}^{(0)}\right) = \vec{x}^{(0)^T}\hat{\vec{\theta}} \qquad \text{and} \qquad \hat{\vec{\theta}} = (X^T X)^{-1} X^T \vec{y}$$

Then,

$$[Bias]^2\left(\hat{y}^{(0)}\right) = 0$$

$Var_D\left(\hat{y}^{(0)}\right)$ achieves the minimum variance among all unbiased (bias=0) estimators.

However, note that this does not mean that the EPE above is the minimum over all predictors!

To lower the total $\mathrm{EPE}\left(\vec{\boldsymbol{x}}^{(\mathbf{0})}\right)$, we must allow and sometimes encourage bias!

More Examples of models:

The other methods will be intermediate between these two 'extremes'.

Examples of intermediate methods would include

(a) Kernel methods (local smoothing)

$$\hat{y} = \hat{h}(\vec{x}) = \sum_{i=1}^{n} w\left(\vec{x} - \vec{x}^{(i)}\right) \cdot y^{(i)}$$

with some fixed local kernel function $w(z)$.

The width and height of $w(z)$ can be considered a 'smoother' version of the question of whether we are in/out of a 'nearest neighborhood'.

For example, the locally weighted linear regression.

(b) Basis function methods.

$$\hat{y} = \hat{h}(\vec{x}) = \sum_{m=1}^{M} \beta_m \cdot h_m(\vec{x})$$

where $h_m(\vec{x})$ are (fixed) basis functions. Note here the β_m are pre-determined and do not depend on $\vec{x}$.

Questions: What is the choice of basis $h_m(\vec{x})$? What is the selection of the basis elements; how do we estimate the β_m?

Examples: $h_m(\vec{x})$ is linear, or polynomial, or Gaussian, or sigmoid, or tanh.

More general basis methods: spline functions (Hastie et al, 2001)

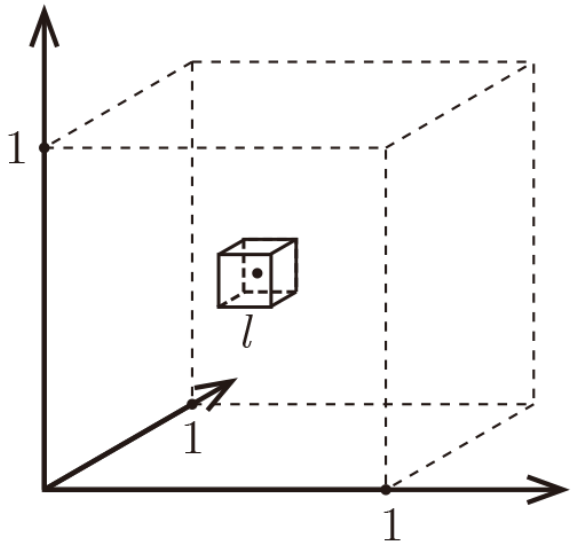
General issues that need to be discussed here include:

1. Dimensionalities of the models, i.e., how many parameters we have.

2. Bias-variance tradeoff, i.e. how to balance systematic vs. random error.

➢ Curse of Dimensionality

Relatively easy to do things in low dimension, but in higher dimension there is a lot of room for things to happen. Data points become sparse in high dimension; more distant from each other.



$$l = \left(\frac{k}{n}\right)^{1/d}$$

d	l
2	0.1
10	0.63
100	0.955
1000	0.9954

$k = 10$

Considering the k nearest neighbors of such a test point $\vec{x}^{(0)}$.
We are no longer very 'local' in high dimension $\mathbb{R}^d$. So as d large enough, almost the entire space is needed to find the 10-NN.

Note the bias (systematic averaged error) is large if we are sampling far-away points from test point $\vec{x}^{(0)}$.

We cannot simply make very small l and only sample a few 'neighbors' of $\vec{x}^{(0)}$.

This would reduce the bias. However, it would raise the variance (variability) our estimate based on only a few points from the training set.

➢ Dimension reduction

It would be better if we would vary the neighborhood 'resources' in lower dimension.

For example, choosing a subset of axes only along the dimensions necessary (i.e. lowering the dimension). Could do this with **feature** (dimension) **selection**. (Subset section or random forests.)

Another way is to find combination of features: For example, SVD/ PCA/LDA embedding to a lower dimension with largest variance of the data.

More advanced methods including, spectral embedding, T-distributed stochastic neighbor embedding, manifold embedding, etc.

References:

- Textbooks:

  Hastie: Sec 3.2, 3.4, and 3.6

  Bishop: Sec 1.4, 3.1, 3.2,

- Online resources

  My 7243 lecture notes.